Cover inequalities for a vehicle routing problem with time windows and shifts

Dabia, Said; Ropke, Stefan; Van Woensel, Tom

published in
Transportation Science
2019

DOI (link to publisher)
10.1287/trsc.2018.0885

document version
Publisher's PDF, also known as Version of record

Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Cover Inequalities for a Vehicle Routing Problem with Time Windows and Shifts

Said Dabia, Stefan Ropke, Tom van Woensel

To cite this article:
Said Dabia, Stefan Ropke, Tom van Woensel (2019) Cover Inequalities for a Vehicle Routing Problem with Time Windows and Shifts. Transportation Science 53(5):1354-1371. https://doi.org/10.1287/trsc.2018.0885

http://pubsonline.informs.org/journal/trsc

TRANSPORTATION SCIENCE
Vol. 53, No. 5, September–October 2019, pp. 1354–1371
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

Cover Inequalities for a Vehicle Routing Problem with Time Windows and Shifts

Said Dabia,[a,b] Stefan Ropke,[c] Tom van Woensel[d]

[a] School of Business and Economics, VU University Amsterdam, 1081 HV Amsterdam, Netherlands; [b] Eyefreight B.V., 3981 AJ Bunnik, Netherlands; [c] Department of Management Engineering, Technical University of Denmark, 2800 Copenhagen, Denmark; [d] School of Industrial Engineering, Eindhoven University of Technology, 5612 AZ Eindhoven, Netherlands

Contact: s.dabia@vu.nl, http://orcid.org/0000-0003-0259-0684 (SD); ropke@dtu.dk (SR); t.v.woensel@tue.nl, http://orcid.org/0000-0003-4766-2346 (TvW)

Received: August 27, 2016
Revised: April 1, 2018; August 26, 2018
Accepted: November 2, 2018
Published Online in Articles in Advance: August 27, 2019

https://doi.org/10.1287/trsc.2018.0885

Abstract. This paper introduces the vehicle routing problem with time windows and shifts (VRPTWS). At the depot, several shifts with nonoverlapping operating periods are available to load the planned trucks. Each shift has a limited loading capacity. We solve the VRPTWS exactly by a branch-and-cut-and-price algorithm. The master problem is a set partitioning with an additional constraint for every shift. Each constraint requires the total quantity loaded in a shift to be less than its loading capacity. For every shift, a pricing subproblem is solved by a label-setting algorithm. Shift capacity constraints define knapsack inequalities; hence we use valid inequalities inspired from knapsack inequalities to strengthen the linear programming relaxation of the master problem when solved by column generation. In particular, we use a family of tailored robust cover inequalities and a family of new nonrobust cover inequalities. Numerical results show that nonrobust cover inequalities significantly improve the algorithm.

Funding: The research of S. Dabia is funded by Dinalog (Project: Bundling at the Source).
Supplemental Material: The online appendix is available at https://doi.org/10.1287/trsc.2018.0885.

Keywords: vehicle routing problem • column generation • shift capacity • branch-and-cut-and-price algorithm

1. Introduction

In the vehicle routing problem with time windows (VRPTW), a homogeneous fleet of vehicles with limited capacity delivers goods to a set of geographically scattered customers. Each customer requires the delivery of a certain amount of goods within a specified time window. The objective of the problem is to determine a set of routes that minimizes the total operational cost while ensuring that all customers are served, that time windows are respected, and that the capacity limit of the vehicles is not violated. It is assumed that all vehicles start and end their routes at a common depot and that travel cost and travel time between each pair of locations in the problem are known.

Because of its practical relevance, the VRPTW is extensively studied in the literature (see, e.g., Gendreau and Tarantilis 2010; Baldacci, Mingozzi, and Roberti 2012; and Desaulniers, Madsen, and Ropke 2014 for some recent surveys). Consequently, many (meta-) heuristics and exact methods are successfully developed to solve this problem. However, most existing models assume that vehicles are simultaneously dispatched at the depot. This is not always the case in real life. In some industries, it is common to let the depot operate all or most of the day and let different work shifts man the depot. Each work shift (e.g., the day, the evening, and the night shifts) has a limited loading capacity. A shift loading capacity is, for instance, the number of full truckloads that can be realized in that shift. Obviously, when the total quantity to be delivered exceeds a shift loading capacity, multiple shifts must be used to load the planned trucks. Because shifts have different start and end times (e.g., the day shift [0700–1500], the evening shift [1500–2300], and the night shift [2300–0700]), some of the planned trucks must be dispatched at a later time. Consequently, the VRPTW model would not be directly applicable in such cases, and the model must be augmented with constraints that ensure that one does not assign too much work to a single shift.

We consider the variant of the VRPTW whereby multiple shifts with limited loading capacity are considered and denote this variant the *VRPTW with shifts* (VRPTWS). The VRPTWS is inspired from a real-life problem faced by a customer (i.e., a freight forwarder) of a software logistics company with which we are working. At the customer's warehouse, three shifts are available, each with a, to be respected, limited loading capacity. We divide the depot's operating period (e.g., a day) into several nonoverlapping time zones such that a different shift is associated with each of these zones. Consequently, the depot's operating period consists of multiple shifts, each with a start and

end time, and a limited loading capacity. In this paper, we determine the set of routes that minimizes the total distance traveled. Additionally, the assignment of routes to the different shifts must take the shift loading capacity into consideration. Trucks loaded in a particular shift s are all assumed to depart from the depot at a certain time l_s. We do not impose constraints that force the vehicles to return in the same shift (in the test instances considered, all routes are supposed to have returned to the depot at a specific time).

We solve the VRPTWS to optimality using a branch-and-cut-and-price (BCP) algorithm. In a BCP algorithm, the linear relaxation of the master problem in each branch-and-bound node is solved by column generation. In case of the VRPTWS, the master problem of the column generation is a set partitioning with an additional constraint for every shift, requiring its loading capacity to be respected. For every shift, a pricing subproblem, which is an elementary shortest-path problem with resource constraints (ESPPRC), is solved by means of a bidirectional label-setting algorithm. To tighten the linear relaxation of the master problem, we include several tailored valid inequalities defined on the compact variables and several new valid inequalities defined directly on the master variables. Although the former are robust inequalities that can be easily handled in the BCP algorithm, the latter are nonrobust inequalities that are shown to be stronger but increase the complexity of the pricing subproblems. We show how to deal with the nonrobust inequalities in the BCP algorithm, which is in line with recent research on vehicle routing problems (VRPs). Subset–row inequalities introduced by Jepsen et al. (2008) are a good example of nonrobust inequalities that are both strong and tractable at the same time. The developed valid inequalities could be applied to several combinatorial optimization problems in which knapsack inequalities appear in the formulation.

The main contributions of this paper are summarized as follows. First, we introduce a new problem that extends the classic VRPTW by considering shifts-limited loading capacity and that is inspired from a real-life application. Second, we present an exact solution based on a BCP algorithm. For every shift, a separate pricing subproblem is solved by means of a bidirectional label-setting algorithm. By exploiting the structure of the problem, we develop new nonrobust valid inequalities to strengthen the linear programming (LP) relaxation of the master problem when solved by column generation. The added valid inequalities are shown to be useful when solving the VRPTWS and could be used in the solution of related problems in which knapsack inequalities are part of the formulation. Moreover, we show how to deal with the additional complexity resulting from including the nonrobust valid inequalities.

The paper is organized as follows. Section 2 reviews the literature relevant to our problem. In Section 3, a formal description of the problem along with its arc flow formulation is provided. In Section 4, the column-generation algorithm is described. Section 5 introduces the valid inequalities used in the BCP framework, and Section 6 explains the separation of the nonrobust cover inequalities. In Section 7, we show how the nonrobust valid inequalities are handled in the pricing subproblems. Section 8 describes the branching decisions. In Section 9, extensive numerical experiments are conducted. Finally, Section 10 concludes the paper.

2. Literature Review

This nonexhaustive literature review deals with two broad topics. We discuss the relevant literature both from an application point of view and from a related methodology point of view. For both cases, our paper significantly adds to the mentioned literature.

An abundant number of publications are devoted to the vehicle routing problem (see Laporte (1992, 2007) and Toth and Vigo (2014) for some reviews). For good reviews on the VRPTW, the reader is referred to Bräysy and Gendreau (2005a, b), Kallehauge (2008), Gendreau and Tarantilis (2010), and Desaulniers, Madsen, and Ropke (2014). Column generation is successfully implemented for several combinatorial optimization problems. For an overview of column-generation algorithms, the reader is referred to Lübbecke and Desrosiers (2005). Column generation in the context of the VRPTW was first introduced by Desrochers et al. (1992). Later, Kohl et al. (1999) introduced subtour elimination constraints and two-path cuts into the column-generation approach, and Cook and Rich (1999) applied the more general k-path cuts. In the 1990s, the pricing subproblem of choice was the shortest-path problem with resource constraints and two-cycle elimination. In Irnich and Villeneuve (2006), an algorithm for k-cycle elimination was introduced, which led to tighter bounds, and Feillet et al. (2004) and Chabrier (2006) proposed algorithms for the ESPPRC that further improved the lower bounds. Righini and Salani (2006, 2008) proposed various techniques to speed up the ESPPRC algorithm, including bidirectional search and decremental state-space relaxation. Jepsen et al. (2008) further improved the lower bounds by proposing a column-generation algorithm with nonrobust valid inequalities called *subset–row inequalities* and showed how to efficiently deal with these nonrobust inequalities in their BCP framework. To accelerate the pricing subproblem solution, Desaulniers, Lessard, and Hadjar (2008) proposed a tabu search heuristic for the ESPPRC.

Furthermore, elementarity was relaxed for a subset of nodes, and both two-path and subset–row inequalities were used. Baldacci, Mingozzi, and Roberti (2011b) introduced a new route relaxation called *ng-route* used to solve the pricing subproblem. Their framework proved to be very effective in solving difficult instances of the VRPTW with wide time windows because they were able to solve all but one of the 56 famous Solomon instances.

In this paper, we apply two types of valid inequalities inspired from cover inequalities for knapsack problems. First, we include *robust cover* inequalities defined on the compact problem variables. These inequalities were first discovered separately by Balas (1975) and Wolsey (1975). We also include a strengthened version of these inequalities, that is, the lifted robust cover inequalities (see, e.g., Gu et al. 1998, Kaparis and Letchford 2008). Second, we include *nonrobust cover* inequalities, which are new valid inequalities defined on the master problem variables. Including nonrobust cover inequalities increases the complexity of the pricing subproblems because each inequality leads to an additional resource. As in Jepsen et al. (2008), we show how to make the newly introduced nonrobust cover inequalities tractable when implemented in a BCP algorithm. The introduced nonrobust cover inequalities could be applied to several combinatorial optimization problems when solved by column generation and when knapsack constraints are part of the set-partitioning formulation. Some examples are the capacitated location routing problem (Baldacci et al. 2011a) and the more general two-echelon capacitated vehicle routing problem (Baldacci et al. 2013), in which a depot's capacity is modeled as a knapsack constraint. In Muter, Cordeau, and Laporte (2014), a branch-and-price algorithm is used to solve the multidepot VRP with interdepot routes, where vehicles are allowed to stop at any depot to replenish and continue with another route. A set of routes traversed by a vehicle is called a *rotation*. The rotation duration must not exceed a maximum D; hence the total duration of the routes included in a rotation is bounded by D. This is again modeled by a knapsack constraint in the set-partitioning formulation.

Closely related to the VRPTWS, Gromicho et al. (2012) consider a combination of vehicle routing and loading dock scheduling, including synchronized routing. Examples of physical constraints mentioned in their paper include a limited number of loading docks and a limited size of loading crews. Additionally, time windows and obedience of compulsory working-time directives are considered as well. This problem is solved using a heuristic-based column generation. Cases obtained from two large retailers are used to demonstrate the value of their approach. These cases also dealt with a heterogeneous fleet with different dock capacity constraints. Ren, Dessouky, and Ordóñez (2010) consider a VRPTW with multiple shifts and overtime. Their problem was inspired by a routing problem in healthcare, in which the vehicles continuously operate in shifts, and overtime is allowed. They introduced a shift-dependent tabu search-based heuristic that takes overtime into account in the routing. The authors developed lower bounds by solving the LP relaxation of a mixed-integer programming (MIP) model with a number of specialized cuts. These cuts give improved bounds on the minimum number of required routes but also give insights on the minimum overtime needed and aim at eliminating two-node cycles.

There are also similarities between the VRPTWS and the multidepot VRP in which the multiple shifts are analogous to the multiple depots. However, in the multidepot VRP, there is traditionally no limit on the total amount of goods that can be shipped from each depot, whereas sometimes there is a limit on the number of vehicles available at each depot. Recent exact methods for the multidepot VRP are described by Baldacci and Mingozzi (2009), Bettinelli, Ceseli, and Righini (2011), and Contardo and Martinelli (2014), whereas a selection of heuristics for the problem are described by Cordeau, Gendreau, and Laporte (1997), Pisinger and Ropke (2007), Ho et al. (2008), Cordeau and Maischberger (2012), and Vidal et al. (2012). It is also relevant to mention the multiperiod VRP (Chao, Golden, and Wasil 1995, Mourgaya and Vanderbeck 2007, Hemmelmayr, Doerner, and Hartl 2009) because it has been shown that a multidepot VRP instance is easily transformed into a multiperiod VRP instance (see Cordeau, Gendreau, and Laporte 1997). Finally, we would like to mention the location–routing problem studied by, for example, Wu, Low, and Bai (2002), Baldacci, Mingozzi, and Calvo (2011a), and Prodhon and Prins (2014). The problem is similar to the multidepot VRP, but in the location–routing problem one determines which depots, from a set of available depots, should be active (at a fixed cost). In relation to the VRPTWS, the location–routing problem is interesting because most models consider a limitation on the amount of goods that can be shipped from any depot, which is similar to the constraint on the capacity that each shift can manage.

3. Problem Description

Consider a graph $G = (V, A)$, where $V = \{0, 1, \ldots, n, n+1\}$ is the set of nodes and $V_c = V \setminus \{0, n+1\}$ represents the set of customers; nodes 0 and $n+1$ represent the depot, and the two nodes are the start and end, respectively, of any route. Let $[a_i, b_i]$ be the delivery time window of node i. A vehicle can arrive at node i before the opening time a_i and wait to start service. Let d_i be the demand and s_i be the service time of node $i \in V$.

We assume, without loss of generality, that $s_0 = s_{n+1} = d_0 = d_{n+1} = a_0 = 0$. Let τ_{ij} and c_{ij} denote the travel time (it includes service time at i) and the travel cost, respectively, from node i to node j. We consider an unlimited fleet of homogeneous vehicles K, each having a finite capacity Q. We can now define the set of feasible arcs as $A = \{(i,j) \in V \times V : i \neq j \wedge a_i + \tau_{ij} \leq b_j \wedge d_i + d_j \leq Q\}$. Furthermore, we assume that an operating period at the depot consists of a set of shifts S. Each shift $s \in S$ has a start time l_s, end time u_s, and a limited loading capacity L_s. We assume that vehicles planned in shift s can be dispatched at time l_s.

We present an MIP arc flow formulation based on the flow variables $x^s_{ijk}, s \in S, k \in K, (i,j) \in A$, that take the value 1 if and only if arc (i,j) is traversed by vehicle k that is loaded in shift s and the time variables $\omega^s_{ik}, s \in S, k \in K, i \in V$, representing the start time of service at node i. Furthermore, for every subset $A' \subseteq A$, vehicle $k \in K$, and shift $s \in S$, we denote $x^s_k(A') = \sum_{(i,j)\in A'} x^s_{ijk}$, and we let $\gamma^+(i)$ and $\gamma^-(i)$ be the set of arcs originating from i and the set of arcs ending in i, respectively. The arc flow formulation of the VRPTWS is as follows:

$$\min z = \sum_{s\in S}\sum_{k\in K}\sum_{(i,j)\in A} c_{ij} x^s_{ijk} \tag{1}$$

subject to

$$\sum_{s\in S}\sum_{k\in K} x^s_k(\gamma^+(i)) = 1 \qquad \forall i \in V_c, \tag{2}$$

$$\sum_{k\in K}\sum_{i\in V} d_i x^s_k(\gamma^+(i)) \leq L_s \qquad \forall s \in S, \tag{3}$$

$$x^s_k(\gamma^+(0)) = x^s_k(\gamma^-(n+1)) = 1 \qquad \forall s \in S, \forall k \in K, \tag{4}$$

$$x^s_k(\gamma^+(i)) = x^s_k(\gamma^-(i)) \qquad \forall s \in S, \forall k \in K, \forall i \in V_c, \tag{5}$$

$$x^s_{ijk}\left(\omega^s_{ik} + \tau_{ij}\right) \leq \omega^s_{jk} \qquad \forall s \in S, \forall k \in K, \forall (i,j) \in A, \tag{6}$$

$$a_i \leq \omega^s_{ik} \leq b_i \qquad \forall s \in S, \forall k \in K, \forall i \in V_c, \tag{7}$$

$$l_s \leq \omega^s_{0k} \leq u_s \qquad \forall s \in S, \forall k \in K, \tag{8}$$

$$\sum_{i\in V} d_i x^s_k(\gamma^+(i)) \leq Q \qquad \forall s \in S, \forall k \in K, \tag{9}$$

$$w^s_{ik} \geq 0 \qquad \forall s \in S, \forall k \in K, \forall i \in V, \tag{10}$$

$$x^s_{ijk} \in \{0,1\} \qquad \forall s \in S, \forall k \in K, \forall (i,j) \in A. \tag{11}$$

The objective function (1) expresses the total cost to be minimized. Constraints (2) ensure that every customer is assigned to exactly one vehicle. Constraints (3) guarantee that shift loading capacity is respected. Constraints (4) and (5) are related to the flow of arcs on the path traversed by a vehicle $k \in K$ that is loaded in shift $s \in S$. Furthermore, constraints (6)–(8) guarantee feasibility with respect to time considerations. Constraints (9) make sure that the vehicles' capacity is respected. Finally, constraints (10) ensure that the time variables are nonnegative, and constraints (11) impose binary conditions on the flow variables.

4. Set Partitioning Formulation and Column Generation

To derive the set-partitioning formulation for the VRPTWS, we define Ω^s as the set of feasible paths corresponding to shift $s \in S$. For a given shift, a path is feasible if it is loaded within the shift operating period and satisfies customers' delivery time windows and vehicle and shift capacity constraints. We let $\Omega = \cup_{s\in S}\Omega^s$ denote the set of all feasible paths. For each path $p \in \Omega$, c_p denotes its cost (i.e., the total distance traveled) and m_p its respective load. Let σ_{ip} be a constant that counts the number of times node i is visited by path p. Furthermore, if y_p is a binary variable that takes the value 1 if and only if the path p is included in the solution, the VRPTWS is formulated as the following set-partitioning problem:

$$\min \sum_{p\in\Omega} c_p y_p \tag{12}$$

subject to

$$\sum_{p\in\Omega} \sigma_{ip} y_p = 1 \qquad \forall i \in V_c, \tag{13}$$

$$\sum_{p\in\Omega^s} m_p y_p \leq L_s \qquad \forall s \in S, \tag{14}$$

$$y_p \in \{0,1\} \qquad \forall p \in \Omega. \tag{15}$$

The objective function (12) minimizes the cost of the chosen routes. Constraints (13) guarantee that each node is visited exactly once. Constraints (14) ensure that the shift loading capacities are respected. We use column generation to solve the LP relaxation of (12)–(15): starting with a small subset of variables, we generate additional variables for the master problem by solving, for each shift $s \in S$, a pricing subproblem that searches for variables with negative reduced cost. Let $\pi_i \in \mathbb{R}$ $(\pi_i \geq 0)$, $i \in V_c$, be the dual variables associated with constraints (13) and $\mu_s \in \mathbb{R}$ $(\mu_s \leq 0)$, $s \in S$, the dual variables associated with constraints (14). The reduced cost of a variable (path) $p \in \Omega^s$ is defined as

$$\bar{c}^s_p = c_p - \sum_{i\in V_c} \sigma_{ip}\pi_i - m_p\mu_s. \tag{16}$$

The dual variable μ_s is negative and therefore will be acting as a penalty when subtracted from the path's reduced cost. If we let x_{ij}^p be a binary variable that takes the value 1 if and only if arc (i,j) is used in path p, the path's load m_p can be expressed as

$$m_p = \sum_{(i,j)\in A} d_i x_{ij}^p. \tag{17}$$

Hence, the reduced cost of path p is expressed as follows:

$$\bar{c}_p^s = \sum_{(i,j)\in A} (c_{ij} - \pi_i - d_i\mu_s)x_{ij}^p. \tag{18}$$

For an overview of column-generation algorithms, the reader is referred to Lübbecke and Desrosiers (2005) and Desaulniers, Desrosiers, and Solomon (2005).

5. Cover Inequalities

Cover inequalities are well-known valid inequalities for the knapsack problem. The 0/1-knapsack polytopes defined by the shift capacity constraints (3) and (14) include the polytopes defined by the compact formulation (1)–(11) and the master problem (12)–(15), respectively. Consequently, any valid inequality for the former is also a valid inequality for the latter. We apply valid inequalities inspired from the knapsack problem to strengthen the LP relaxation of the master problem when solved by column generation. We introduce a family of tailored cover inequalities defined on the compact variables and a family of new valid inequalities defined directly on the master variables. We call cover inequalities expressed in the compact variables *robust cover inequalities*; cover inequalities expressed in the master variables are called *nonrobust cover inequalities*.

5.1. Robust Cover Inequalities

For every shift $s \in S$, the corresponding shift capacity constraint (3), along with the flow variables $\mathbf{x}^s = \{x_a^s : a \in A\}$ of the compact formulation (1)–(11), defines the 0/1-knapsack structure

$$X_s = \left\{\mathbf{x}^s \in \mathbb{B}^{|A|} : \sum_{a\in A} d_a x_a^s \le L_s\right\}, \tag{19}$$

in which the items are the arcs in A, the weight d_a of each arc $a = (i,j) \in A$ is the demand d_i of its start node i, and the knapsack capacity is equal to the shift capacity L_s. Therefore, valid inequalities for the convex hull of X_s defined on the compact variables $\mathbf{x}^s$ can be used to strengthen the LP relaxation of the master problem. A subset $C \subseteq A$ is called a *cover* if $\sum_{a\in C} d_a > L_s$. Moreover, C is a *minimal* cover if no proper subset of C is also a cover; that is, for every $a' \in C$, it holds that $\sum_{a\in C\setminus\{a'\}} d_a \le L_s$. For any minimal cover C, the inequality

$$\sum_{a\in C} x_a^s \le |C| - 1 \tag{20}$$

is valid for the convex hull of X_s. It simply says that a subset of customers with a total demand larger than the shift loading capacity cannot all be planned on vehicles loaded in the same shift. It can be extended by the arcs in the set $\bar{C} = \{a \in A \setminus C : d_a \ge \tau\}$, where $\tau = \max\{d_a : a \in C\}$ is called the *inequality threshold*. Hence, the inequality

$$\sum_{a\in C\cup\bar{C}} x_a^s \le |C| - 1 \tag{21}$$

is also valid for the convex hull of X_s and is called a *robust cover inequality*.

5.1.1. Separation of Robust Cover Inequalities.

For a given shift $s \in S$ and its corresponding fractional solution $\mathbf{x}^{*s}$, the separation of the robust cover inequalities (21) implies finding a subset of arcs C (i.e., a cover) such that the total quantity delivered on these arcs exceeds the shift capacity L_s, and $\sum_{a\in C} x_a^{*s} > |C| - 1$. Introducing the binary variable z_a^s that takes the value 1 if and only if arc a is included in the cover C, the separation problem for the robust cover inequalities is equivalent to

$$\kappa = \min\left\{\sum_{a\in A}(1 - x_a^{*s})z_a^s : \sum_{a\in A} d_a z_a^s > L_s\right\}. \tag{22}$$

A violated robust cover inequality is found if and only if $\kappa < 1$. The separation problem (22) is equivalent to a knapsack problem and can be solved by dynamic programming, as described in Pisinger (1997).

5.2. Lifted Robust Cover Inequalities

Inequalities (20) can also be strengthened by lifting up the variables corresponding to the arcs in $A \setminus C$ and adding them to the left-hand side of the inequalities, which results in the robust cover inequalities

$$\sum_{a\in C} x_a^s + \sum_{a\in A\setminus C} \alpha_a x_a^s \le |C| - 1, \tag{23}$$

where the nonnegative integers α_a are as large as possible. In this paper, we use simple heuristics as described in Gu et al. (1998) and Kaparis and Letchford (2008) to generate violated lifted robust cover inequalities (23).

We denote $\mathscr{CC}$ the set of robust cover inequalities (21) and (23) added to the LP relaxation of the master problem.

5.3. Nonrobust Cover Inequalities

In this section, we introduce new families of nonrobust valid inequalities for the VRPTWS defined directly on the master problem variables.

5.3.1. Ordinary Nonrobust Cover Inequalities. Similar to Section 5.1, we define nonrobust cover inequalities on the y variables. In this case, a cover is a subset $C \subseteq \Omega^s$ such that $\sum_{p\in C} m_p > L_s$. For any minimal cover C, the inequality

$$\sum_{p\in C} y_p \leq |C| - 1 \tag{24}$$

is valid and is denoted a *nonrobust y-cover inequality* (or simply *y-cover inequality*) in the following. As in Section 5.1, we define the threshold $\tau = \max\{m_p : p \in C\}$, set $\bar{C} = \{p \in \Omega^s \setminus C : m_p \geq \tau\}$, and define the *extended y-cover inequality*

$$\sum_{p\in C\cup\bar{C}} y_p \leq |C| - 1, \tag{25}$$

which is a strengthening of (24).

Unfortunately, the y-cover and extended y-cover inequalities are not very useful in practice. Let us first consider the y-cover inequalities (24). They avoid that a set of variables C is used together in a feasible solution. For shift s, the set C has in general very low cardinality compared with the entire set of variables Ω^s. Consequently, even though the LP solution of the master problem is going to change after applying a y-cover inequality, it is likely that the new solution will be similar to the previous one (because there are so many, almost identical, variables to choose from in Ω^s). This means that many inequalities have to be added before the LP solution changes significantly. Furthermore, it is complicated to handle the constraints in the pricing subproblem. Each constraint is associated with a dual variable v that has to be subtracted from the reduced cost in case the pricing ends up constructing a path from C. This complicates the use of dominance rules in a label-setting algorithm, and the difficulties worsen as more constraints are added to the master problem.

The extended y-cover inequalities are stronger because they include the set $\bar{C}$ on the left-hand side, and that set can have a considerable size. The extension does not further complicate the pricing subproblem. However, the difficulties in keeping track of the paths from C in the pricing subproblem persist. We implemented both versions of the inequality, but the results were not encouraging. In the following, we present stronger inequalities rooted in the y-cover inequalities.

5.3.2. Nonrobust *k*-Cover Inequalities. In this section, we introduce a family of simple nonrobust cover inequalities we call the *nonrobust k-cover inequalities* (or simply *k-cover inequalities*). For shift $s \in S$ and integer $k \geq 1$, we define a k-cover

$$C = \left\{p \in \Omega^s : m_p > \frac{L_s}{k}\right\} \tag{26}$$

as the subset of paths with a load larger than the threshold $\tau = \frac{L_s}{k}$. Now we can define the k-cover inequalities as follows.

Definition 1. For shift $s \in S$, consider the k-cover C for some $k \geq 1$. The k-cover inequalities are defined as

$$\sum_{p\in C} y_p \leq k - 1. \tag{27}$$

The validity of the k-cover inequalities is formulated in the following proposition.

Proposition 1. *The k-cover inequalities* (27) *are valid for problem* (12)–(15).

Proof. Let C be a k-cover defined on shift s. A solution that violates inequality (27) uses at least k routes from C. Because each path in C has a load larger than $\frac{L_s}{k}$, the solution violates the shift's capacity L_s. □

Example 1. Consider the fractional solution in Table 1 obtained after solving the master problem for an instance of 25 customers and three shifts, each with loading capacity 200, and after adding the robust cover inequalities found by our separation methods. The first column shows the paths' indices, the second column corresponds to the paths' weights in the LP solution, the third column shows the shifts in which the paths are planned, the fourth column represents the paths' loads, and the fifth column shows the sequence of a path. Note that paths 5, 6, 9, and 10 have a weight 0 in the current solution but are, however, reported in Table 1 for explanation purposes. For shift $s = 1$ and $k = 2$, paths 7, 8, and 10 all have a load larger than the threshold $\tau = \frac{200}{2} = 100$ and therefore imply the two-cover $C = \{7, 8, 10\}$ that defines the violated two-cover inequality $y_7 + y_8 + y_{10} \leq 1$.

Table 1. Example of a Fractional Solution

p	y_p	s	m_p	Path
1	0.67	0	70	5, 3, 7, 8, 10
2	0.01	0	190	13, 17, 18, 19, 15, 16, 14, 12
3	1.00	0	100	20, 24, 25, 23, 22, 21
4	0.26	0	160	5, 3, 7, 8, 10, 11, 9, 6, 4, 2, 1
5	0.00	0	150	20, 24, 25, 23, 22, 21, 17, 18, 19
6	0.00	0	120	21, 21, 25, 8, 9, 10
7	0.99	1	190	13, 17, 18, 19, 15, 16, 14, 12
8	0.07	1	160	5, 3, 7, 8, 10, 11, 9, 6, 4, 2, 1
9	0.00	1	80	22, 23, 24, 25
10	0.00	1	120	16, 25, 21, 22
11	0.67	2	90	11, 9, 6, 4, 2, 1

We denote $\mathcal{MC}_1$ as the set of k-cover inequalities (27) added to the LP relaxation of the master problem (12)–(15). We will return to the question of how to handle these inequalities in the pricing subproblem in Section 7.

5.3.3. Nonrobust p-Cover Inequalities. In Section 5.3.1, we argued that the impact of the y-cover inequality (24) is weak. In this section, we propose a strengthened form denoted *nonrobust p-cover inequalities* (or simply *p-cover inequalities*). Each p-cover inequality is based on a y-cover inequality.

For shift s, let $V(p)$ be the set of nodes visited along path $p \in \Omega^s$. Furthermore, let us call path p a *super path* of path p' if $V(p') \subseteq V(p)$. Let C be a cover defined as in Section 5.3.1. We *trim* the paths in C by finding and removing the node with the lowest demand from one path in C that visits it, resulting in a new set $\widetilde{C}$. Each time a node is removed, we check whether the inequality is still valid by checking whether $\sum_{p\in\widetilde{C}} m_p > L_s$. If it is not, the trimming procedure backtracks to the last definition of $\widetilde{C}$ that resulted in a valid inequality.

Let $\tau = \min\{m_p : p \in C\}$; we can define $\mathcal{P}(C) = \{p \in \Omega^s : (\exists p' \in \widetilde{C} : V(p') \subseteq V(p)) \vee m_p \geq \tau\}$ as the set of paths that are either a super path of one of the paths in $\widetilde{C}$ or have a load greater than or equal to the threshold τ. Now we can introduce the p-cover inequalities as

$$\sum_{p\in\mathcal{P}(C)} y_p \leq |C| - 1. \tag{28}$$

The validity of the p-cover inequalities is formulated in the following proposition.

Proposition 2. *The p-cover inequalities* (28) *are valid for problem* (12)–(15).

Proof. To see that the inequalities (28) are valid, we first notice that if $p \in \Omega^s$ is a super path of some path $p' \in \widetilde{C}$, then $m_p \geq m_{p'}$. Additionally, only one of the super paths of p' can be used in an integer solution because they all visit all customers in p', and each customer can only be visited once. If an integer solution violates the inequality, it must be using at least $|C|$ paths from $\mathcal{P}(C)$. Let p be a path from $\mathcal{P}(C)$ that is included in this solution. Path p will be included either because its load exceeds τ or because it is a super path of a path from $\widetilde{C}$. If p is a super path of a path p' from $\widetilde{C}$, we will have $m_p \geq m_{p'}$, and p will be the only super path of p' included in the solution. Altogether this means that the solution violates the shift capacity, and therefore, the inequality is valid. □

We should note that trimming C results in stronger inequalities because it leads to more variables being added to the left-hand side of (28). We denote $\mathcal{MC}_2$ as the set of nonrobust p-cover inequalities added to the LP relaxation of the master problem (12)–(15).

Example 2. Consider the fractional solution of Table 1 and customers' demand provided in Table 2. For shift 0, paths 3 and 4 define the minimal cover $C = \{3, 4\}$ that results in the violated y-cover inequality

$$y_3 + y_4 \leq 1. \tag{29}$$

The threshold for inequality (29) is $\tau = 160$. Moreover, the subset of visited nodes on path 3 is $V(3) = \{20, 21, 22, 23, 24, 25\}$, and the subset of visited nodes on path 4 is $V(4) = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$. Every path with a load at least equal to $\tau = 160$ and all super paths of paths 3 and 4 must be added to the left-hand side of inequality (29). Path 2 has load $190 > \tau$, and path 5 is a super path of path 3, therefore, inequality (29) can be strengthened to

$$y_2 + y_3 + y_4 + y_5 \leq 1. \tag{30}$$

The total load of the cover $C = \{3, 4\}$ is 260, and the loading capacity of shift 0 is 200. Therefore, there is room for trimming the subsets $V(3)$ and $V(4)$. Trimming the cover C results in the trimmed cover $\widetilde{C}$ with trimmed paths $\tilde{p}_3 = (21, 22, 25)$ and $\tilde{p}_4 = (5, 7, 8, 10, 11, 9, 6, 2, 1)$ and a total load of 210, which is still larger than the shift capacity 200. Now, for path p to be added to inequality (29), it suffices that $V(\tilde{p}_3) \subseteq V(p)$ or $V(\tilde{p}_4) \subseteq V(p)$. From the paths in Table 1, path 6 is a super path of the trimmed path $\tilde{p}_3$ and therefore can be used to strengthen inequality (30), resulting in the p-cover inequality

$$y_2 + y_3 + y_4 + y_5 + y_6 \leq 1. \tag{31}$$

5.3.4. Nonrobust q-Cover Inequalities. In this section, we introduce a family of valid inequalities, denoted the *nonrobust q-cover inequalities* (or simply *q-cover inequalities*). These inequalities are inspired from the p-cover inequalities by noting that the idea of super paths can be simplified further. For a shift $s \in S$, a customer $i \in V_c$, and an integer $q \geq 1$, we define

$$\Omega^s(i, q) = \{p \in \Omega^s : i \in V(p) \wedge m_p \geq q\} \tag{32}$$

as the subset of paths for shift s that visit customer i and have a load larger than or equal to q. Consider k distinct customers $\mathcal{F} = \{f_1, \ldots, f_k\}$ and k positive integers $\mathcal{Q} = \{q_1, \ldots, q_k\}$ (defined in general regardless

Table 2. Customers' Demand

i	d_i	i	d_i	i	d_i	i	d_i	i	d_i
1	10	6	20	11	10	16	40	21	20
2	30	7	20	12	20	17	20	22	20
3	10	8	20	13	30	18	20	23	10
4	10	9	10	14	10	19	10	24	10
5	10	10	10	15	40	20	10	25	40

of the demands of the customers in $\mathcal{F}$) such that $\sum_{i=1}^{k} q_i > L_s$, and set $q_{\max} = \max_{i=1,\dots,k}\{q_i\}$. We define the set

$$T = \left(\bigcup_{i=1}^{k} \Omega^s(f_i, q_i)\right) \cup \Omega^s(q_{\max}),$$

where $\Omega^s(q_{\max}) = \{p \in \Omega^s : m_p \geq q_{\max}\}$. We can show that at most $k-1$ paths from this set can be visited in a feasible solution that gives rise to the following valid inequality:

$$\sum_{p \in T} y_p \leq k - 1.$$

This inequality can be generalized. Let η be the maximum number of distinct items from $\mathcal{Q}$ that can be packed in a knapsack with capacity L_s (η may be smaller than $k-1$). Then the q-cover inequalities with the threshold $q_{\max}$ are defined as

$$\sum_{p \in T} y_p \leq \eta. \tag{33}$$

The validity of the q-cover inequalities is formulated in the following proposition.

Proposition 3. *The q-cover inequalities (33) are valid for problem (12)–(15).*

Proof. The proof of validity of the q-cover inequalities follows that of the p-cover inequalities. If an integer solution violates a q-cover inequality, it must be using at least $\eta + 1$ paths from T. Let T^* denote the paths from T used in this solution. For any $p \in T^*$, we know that either $m_p \geq q_{\max}$ or $p \in \Omega^s(f_i, q_i)$ for some $i \in \{1, \dots, k\}$. In the latter case, it will be the only path from $\Omega^s(f_i, q_i)$ because all paths in the set $\Omega^s(f_i, q_i)$ visit node f_i, and we know that a customer can only be visited once. Hence, $m_p \geq q_i$. This means that $\sum_{p \in T^*} m_p > L_s$ owing to the definition of $\mathcal{Q}$ and $q_{\max}$, which proves the validity of the inequality. □

We denote $\mathcal{MC}_3$ as the set of nonrobust q-cover inequalities added to the LP relaxation of the master problem (12)–(15). In what follows, we present an example in which no nonrobust k-cover and p-cover inequality is violated, but a violated nonrobust q-cover inequality is found.

Example 3. Consider the fractional solution in Table 3 obtained after solving an instance of 25 customers and three shifts, each with loading capacity of 190, and after adding all violated (lifted) robust cover inequalities. Note that path 10 has weight 0 in the current solution; it is, however, reported for explanation purposes.

For this example, it is not possible to find a violated nonrobust k-cover or p-cover inequality. For shift 0, consider the nonrobust q-cover inequality defined by the sets $\mathcal{F} = \{3, 6, 11, 21\}$ and $\mathcal{Q} = \{48, 38, 45, 62\}$.

Table 3. Example of a Fractional Solution

p	y_p	s	m_p	Path
1	0.33	0	50	7, 11, 19, 10
2	0.33	0	25	8, 10
3	0.33	0	38	18, 6, 13
4	0.02	0	65	14, 16, 6, 13
5	0.33	0	54	7, 11, 8, 17, 5
6	0.67	0	12	18
7	0.33	0	45	11, 19, 10
8	1.00	0	48	3, 9, 20, 1
9	1.00	0	62	21, 23, 24, 12
10	0.00	0	72	14, 13, 23
11	0.65	1	65	14, 16, 6, 13
12	1.00	1	58	15, 22, 4, 25
13	0.33	1	47	16, 17, 5
14	0.33	1	59	7, 19, 8, 17, 5

Demands sum to 193; hence η is equal to 3, and we have that $q_{\max} = 62$. Paths $\{1, 3, 4, 5, 7, 8, 9, 10\}$ define the nonrobust q-cover inequality

$$y_1 + y_3 + y_4 + y_5 + y_7 + y_8 + y_9 + y_{10} \leq 3. \tag{34}$$

The left-hand side takes the value 3.35. Path 10 is added to the inequality (34) because its load is $72 > q_{\max}$.

5.4. Further Robust Cuts from the VRP Literature

In addition to the already-mentioned cuts, we have also implemented two robust cuts known from the capacitated vehicle routing problem (CVRP) and VRPTW literature; these are the *rounded capacity constraints* (see, e.g., Lysgaard, Letchford, and Eglese 2004) and *two-path cuts* (Kohl et al. 1999). After solving the LP relaxation of the set-partitioning problem (12)–(15), we recover x_{ij}^s variables from the y_p variables. We then compute $x_{ij} = \sum_{s \in S} x_{ij}^s$ for all $(i, j) \in A$, and these values are passed to separation routines for the two constraints. The separation algorithm for the rounded capacity constraints use the heuristic described in Lysgaard (2003), whereas for the two-path cuts we use a simple greedy heuristic.

6. Separation of Nonrobust Cover Inequalities

This section explains the separation procedures for the nonrobust cover inequalities. The k-cover inequalities are the easiest to separate because this can be done through a simple and fast enumeration. The separation procedures for the p-cover and q-cover inequalities are more involved.

For the p-cover inequalities, the separation procedure implemented is heuristic. The heuristic first separates a y-cover inequality using a knapsack separation similar to the one described in Section 5.1. If it finds a violated y-cover inequality of the form (24), it will simply construct an inequality as defined by (28), by trimming the paths in C, as explained in Section 5.3.3.

The separation of the q-cover inequalities can be defined as follows: for a shift $s \in S$, given the current LP solution $\mathbf{y}^{*s} = \{y_p^* : p \in \Omega^s\}$, find the set of nodes $\mathcal{F}$ and the set of minimum loads $\mathcal{Q}$ that define the most violated nonrobust q-cover inequality.

First, we use a heuristic separation procedure to detect violated nonrobust q-cover inequalities. The heuristic procedure constructs a nonrobust q-cover inequality based on a, not necessarily violated, y-cover inequality. We construct a cover C defining a y-cover inequality as described in Section 5.3.1 by solving a knapsack problem. From each path $p \in C$, we randomly select a node i and add it to the set of nodes $\mathcal{F}$ and add the load $q_i = m_p$ corresponding to node i to the set of minimum loads $\mathcal{Q}$. Note that if the number of nodes visited by all paths in C is larger than $|C|$, it is always possible to select $|C|$ different nodes. To be able to add more variables to the left-hand side of the nonrobust q-cover inequality, we trim the loads in the set $\mathcal{Q}$. The trimming procedure sorts the loads in $\mathcal{Q}$ and trims the highest ones first as long as the total load in $\mathcal{Q}$ exceeds the shift capacity L_s. The trimming procedure tries to minimize the size of the threshold τ. It also tries to minimize the difference between the highest and the lowest loads in the trimmed set $\mathcal{Q}$.

If the heuristic separation procedure fails to detect a violated nonrobust q-cover inequality, we may use an exact separation procedure. The exact separation procedure formulates and solves a mixed-integer program. Let Ω_i be the set of paths in shift s visiting customer i in the current LP solution, and let $D_i = \left\{q_i^1, q_i^2, \ldots, q_i^{|D_i|}\right\}$ be the set of possible loads to associate with customer i. The set D_i is found by taking the union of the demands of the paths in Ω_i. Furthermore, we define V_s as the set of customers assigned to shift s in the current LP solution, and α_{pk} is 1 if load m_p of path $p \in \Omega_i$ is larger than q_i^k; 0 otherwise. We let z_i be a binary variable that takes value 1 if and only if node i is included in the set $\mathcal{F}$, and we let ξ_{ik} be a binary variable that takes value 1 if and only if load $q_i^k \in D_i$ is associated with node i. Finally, we let x_{ip} be a binary variable that takes value 1 if and only if path p is in the set $\Omega^s(i,q)$, and we let δ_p be a binary variable that takes the value 1 if and only if path p is included in the cover defining the q-cover inequality we are trying to separate. The separation problem is formulated as an MIP as follows:

$$\max \sum_{i \in V_s} \sum_{p \in \Omega_i} y_p^* \delta_p - \sum_{i \in V_s} z_i \tag{35}$$

subject to

$$\sum_{i \in V_s} \sum_{k=1}^{|D_i|} q_i^k \xi_{ik} \geq L_s + 1, \tag{36}$$

$$\sum_{k=1}^{|D_i|} \xi_{ik} \leq z_i \qquad \forall i \in V_s, \tag{37}$$

$$x_{ip} \leq \sum_{k=1}^{|D_i|} \alpha_{pk} \xi_{ik} \qquad \forall i \in V_s, \forall p \in \Omega_i, \tag{38}$$

$$\delta_p \leq \sum_{i \in V_s} x_{ip} \qquad \forall p \in \Omega_i, \tag{39}$$

$$z_i, \xi_{ik}, x_{ip}, \delta_p \in \{0,1\} \qquad \forall i \in V_s, \forall p \in \Omega_i, \forall k \in \{1,2,\ldots,|D_i|\}. \tag{40}$$

The objective function (35) maximizes the violation of the found inequality. The terms $\sum_{i \in V_s} \sum_{p \in \Omega_i} y_p^* \delta_p$ and $\sum_{i \in V_s} z_i$ correspond to the left-hand and right-hand sides, respectively, of the inequality (33). A violated inequality is detected if the objective value is greater than −1. Constraint (36) ensures that the sum of the selected loads is larger than the shift capacity. Constraints (37) ensure that at most one load is selected per customer. Constraints (38) guarantee that a path can be included in the set $\Omega^s(i,q)$ if its load is larger than q. Furthermore, constraints (39) ensure that we can only add a path to the cover defining q-cover inequality we are trying to separate if it is in at least one of the $\Omega(i,q)$ sets.

7. The Label-Setting Algorithm

Each shift defines a pricing subproblem that corresponds to an ESPPRC in which the constrained resources are time and vehicle capacity. Our ESPPRC algorithm is based on a bidirectional label-setting algorithm. First, we use a pricing heuristic in which we limit the number of unprocessed labels. When the heuristic fails to find new paths with negative reduced cost, we call the exact labeling algorithm. Let $p(L)$ be the partial path associated with a label L. The label L is coded using the following attributes:

$v(L)$	Last node visited on the partial path $p(L)$
$c(L)$	Reduced cost of the partial path $p(L)$
$d(L)$	Total quantity delivered along the partial path $p(L)$
$t(L)$	Ready time at node $v(L)$ when reached through the partial path $p(L)$
$V(L)$	Set of nodes visited along the partial path $p(L)$

Furthermore, we denote $\overline{V}(L)$ as the set $V(L)$ extended by the nodes that cannot be visited by label L because of time windows and vehicle capacity. That is,

$$\overline{V}(L) = V(L) \cup \{j \in V_c \setminus V(L) : t(L) + \tau_{v(L),j} > b_j \ \vee \ d(L) + d_j > Q\}.$$

In the labeling algorithm, for every label, all possible extensions are derived and stored. It ends when all labels are processed. However, the number of labels that can be processed is typically very large. To reduce the number of labels, a dominance test is introduced. Let $E(L)$ denote the set of feasible extensions

of the label L to node $n+1$. More formally, $E(L)$ is the set of all partial paths that can depart at node $v(L)$ and reach node $n+1$ without violating time windows, which have total demand of less than $Q-d(L)$ and which do not use nodes from $V(L)$. If $L' \in E(L)$, we denote $L \oplus L'$ as the label resulting from extending L by L'. Dominance is defined as follows.

Definition 2. Label L_2 is dominated by label L_1 if
1. $v(L_1) = v(L_2)$.
2. $E(L_2) \subseteq E(L_1)$.
3. $c(L_1 \oplus L) \leq c(L_2 \oplus L), \forall L \in E(L_2)$.

Definition 2 states that any feasible extension of label L_2 is also feasible for label L_1. Furthermore, extending L_1 should always result in a better route. However, it is not straightforward to verify the conditions of Definition 2 because it requires the computation and evaluation of all feasible extensions of both labels L_1 and L_2. Consequently, sufficient dominance criteria that are computationally less expensive are desirable. Therefore, in Proposition 4, the sufficient conditions 1–5 are introduced.

Proposition 4 (Feillet et al. 2004)**.** *Label L_2 is dominated by label L_1 if*
1. $v(L_1) = v(L_2)$.
2. $c(L_1) \leq c(L_2)$.
3. $t(L_1) \leq t(L_2)$.
4. $d(L_1) \leq d(L_2)$.
5. $\overline{V}(L_1) \subseteq \overline{V}(L_2)$.

7.1. Solving the Modified Pricing Subproblem

Including nonrobust cover inequalities is not straightforward as the pricing becomes more expensive. In particular, the standard dominance test of Proposition 4 cannot be directly used. Considering the fractional solution of Table 1, by applying standard dominance criteria as described in Proposition 4, partial path (0, 20, 24) might dominate partial path (0, 19, 24). However, when extended all the way to the end node, we might have that (0, 19, 24, 25, 23, 22, 21, 0) is a better path than (0, 20, 24, 25, 23, 22, 21, 0) because the latter gets penalized by the nonrobust p-cover inequality (29) dual variable, whereas the former does not. Next, we will focus on how we handle the complications stemming from adding nonrobust cover inequalities in the pricing subproblem.

7.1.1. Handling Nonrobust k-Cover Inequalities.

Nonrobust k-cover inequalities $\mathcal{MC}_1$ are easily handled in the pricing subproblem. For every generated path, we just need to subtract the dual variables corresponding to the nonrobust k-cover inequalities in $\mathcal{MC}_1$ for which the inequality threshold is surpassed by the path's load when the end node is reached. Furthermore, we can use standard dominance test, as described in Proposition 4. Condition 4 ensures that if any extension of label L_1 by some label L into a path that must be added to a nonrobust k-cover inequality in $\mathcal{MC}_1$, extending L_2 by L must be added to the same inequality. In fact, if the load of path $p(L_1 \oplus L)$ surpasses the inequality threshold, the load of path $p(L_2 \oplus L)$ must surpass the inequality threshold as well. So dominance does not have to know about all the paths in the nonrobust k-cover inequalities $\mathcal{MC}_1$.

Example 4. Let's consider again the fractional solution in Table 1. For shift $s=1$ and integer $k=2$, the two-cover $C=\{7,8,10\}$ defines the nonrobust two-cover inequality with threshold $\tau_2 = 100$, depicted by the equation $y_7 + y_8 + y_{10} \leq 1$. Furthermore, consider two labels L_1 and L_2 such that $p(L_1) = (20, 21, 16)$ and $d(L_1) = m_{p(L_1)} = 70$ and $p(L_2) = (14, 15, 16)$ and $d(L_2) = m_{p(L_2)} = 90$. Moreover, we have that $\overline{V}(L_1) = V(L_1)$ and $\overline{V}(L_2) = V(L_2) \cup \{10, 20, 21\}$. Let L be an extension of label L_1 such that $d(L) = 40$. The total demand of the extended label $L_1 \oplus L$ is $d(L_1 \oplus L) = 110 > \tau_2$; hence path $p(L_1 \oplus L)$ must be added to the two-cover C. Obviously, $p(L_2 \oplus L)$ must be added to C as well because $d(L_2 \oplus L) = 130 > \tau_2$. In other words, for any label L, it will never happen that $p(L_1 \oplus L)$ will be penalized by the dual variable corresponding to the two-cover inequality defined by C and $p(L_2 \oplus L)$ will not. Therefore, condition 2 of the standard dominance test of Proposition 4 is still handling label cost correctly.

7.1.2. Handling Nonrobust p-Cover and q-Cover Inequalities.

For every valid nonrobust inequality $I \in \mathcal{MC}_2 \cup \mathcal{MC}_3$, we need to ensure that its dual variable is subtracted from the reduced cost of a path p that contributes to its violation. This is easily done by checking whether the path's load m_p surpasses the inequality threshold (i.e., τ if $I \in \mathcal{MC}_2$ and q_{max} if $I \in \mathcal{MC}_3$). Moreover, if I defines a nonrobust p-cover inequality, p contributes to the violation of I if it is a super path of a path in the cover defining I. If I defines a nonrobust q-cover inequality, p contributes to the violation of I if it is in one of the subsets $\Omega^s(i,q)$ used to construct the cover defining I. The complexity comes in the dominance test, where we have to account for the possibility that one of the labels that needs to be compared might contribute to the violation of I and the other might not. Next, we will discuss the impact of including nonrobust cover inequalities in $\mathcal{MC}_2 \cup \mathcal{MC}_3$ on the dominance criterion. We differentiate between two cases.

***Case* 1.** For any label L, the elementarity constraint in the pricing subproblem is handled through the set of visited nodes $V(L)$; the standard dominance test will require that $V(L_1) \subseteq V(L_2)$ if label L_1 should dominate label L_2. This condition, together with condition 4 of the dominance test of Proposition 4, is sufficient for handling nonrobust cover inequalities in $\mathcal{MC}_2 \cup \mathcal{MC}_3$. In fact, if L is a feasible extension of L_1 such that $p(L_1 \oplus L)$ contributes to the violation of a nonrobust cover

inequality $I \in \mathcal{MC}_2 \cup \mathcal{MC}_3$, extending L_2 by the same extension L will imply that path $p(L_2 \oplus L)$ contributes to the violation of I as well. In fact, if $p(L_1 \oplus L)$ is active in I because its load surpasses its threshold, condition 4 of Proposition 4 will force $p(L_2 \oplus L)$ to be active in I. If $I \in \mathcal{MC}_2$ and $p(L_1 \oplus L)$ is active in I because it is a super path of some path p' in the cover defining I, then condition 5 of Proposition 4 ensures that $p(L_2 \oplus L)$ is a super path of p' and hence must be active in I. Furthermore, if $I \in \mathcal{MC}_3$ and $p(L_1 \oplus L)$ is active in I because $p(L_1 \oplus L) \in \Omega^s(i,q)$ for some $i \in V_c$ and integer q, then conditions 4 and 5 of Proposition 4 imply that $p(L_2 \oplus L) \in \Omega^s(i,q)$, and hence, $p(L_2 \oplus L)$ must be active in I. Therefore, the dominance criterion will be similar to the one in Proposition 4, with the only difference being that condition $\overline{V}(L_1) \subseteq \overline{V}(L_2)$ must be relaxed to $V(L_1) \subseteq V(L_2)$.

***Case* 2.** Elementarity is handled by keeping track of the nodes that cannot be visited by a label L (i.e., using the set $\overline{V}(L)$); then we need more information to do the dominance test correctly. In fact, we need to keep the set of nodes that are visited by the partial path $p(L)$ to be able to judge whether an extension of label L might lead to a path that must be included in a nonrobust cover inequality in $\mathcal{MC}_2 \cup \mathcal{MC}_3$, and we must subtract the corresponding dual variable from the reduced cost of the partial path $p(L)$. If we consider label L_2 as described in Example 4, it is not possible, knowing only $\overline{V}(L_2)$, to judge whether an extension of L_2 might, in the worst case, lead to a path that contributes to some nonrobust cover inequality in $\mathcal{MC}_2 \cup \mathcal{MC}_3$. For labels L_1 and L_2 of Example 4 and the fraction solution in Table 1, it is clear that, in the worst case, label L_1 might be extended to a path that must be penalized by the dual of the p-cover inequality defined by the cover $C = \{3,4\}$. In fact, the partial path $p(L_1)$ has already visited customers 20 and 21 that are also visited by path 3.

Let us consider a nonrobust cover inequality $I \in \mathcal{MC}_2 \cup \mathcal{MC}_3$. If $I \in \mathcal{MC}_2$, we denote $\widetilde{C}_I$ as the *trimmed cover* and τ_I as the *threshold* corresponding to I, as described in Section 5.3.3. If $I \in \mathcal{MC}_3$, we denote $\mathcal{F}_I$ and $\mathcal{Q}_I$ as the sets used to construct I, as described in Section 5.3.4. Moreover, we let $v_I < 0$ be the dual of I. For a subset of customer nodes $\mathcal{N} \subseteq V_c$, we let

$$\phi(\mathcal{N}) = \left\{ I \in \mathcal{MC}_2 : \left(\bigcup_{p \in \widetilde{C}_I} V(p) \right) \cap \mathcal{N} \neq \emptyset \right\} \cup \{ I \in \mathcal{MC}_3 : \mathcal{F}_I \cap \mathcal{N} \neq \emptyset \}$$

be the subset of nonrobust cover inequalities in $\mathcal{MC}_2 \cup \mathcal{MC}_3$ that "use a node from $\mathcal{N}$." The dominance test can now be written as follows.

Proposition 5. *Label L_2 is dominated by label L_1 if*

1. $v(L_1) = v(L_2)$.
2. $c(L_1) - \sum_{I \in \phi(V(L_1) \setminus V(L_2))} v_I \leq c(L_2)$.
3. $t(L_1) \leq t(L_2)$.
4. $d(L_1) \leq d(L_2)$.
5. $\overline{V}(L_1) \subseteq \overline{V}(L_2)$.

The idea of condition 2 in the dominance test of Proposition 5 is that we, in the worst case, need to subtract all the dual variables corresponding to the nonrobust cover inequalities in $\mathcal{MC}_2 \cup \mathcal{MC}_3$ that are active in the extension of label L_1 but not in the extension of label L_2. These inequalities are included in the set $\phi(V(L_1) \setminus V(L_2))$.

The dominance test can be further improved because we can determine that some of the nonrobust cover inequalities in $\mathcal{MC}_2 \cup \mathcal{MC}_3$ will never be active for a given path. Furthermore, we can also determine that some inequalities will for sure be active for any extension of label L_2. Let

$$\chi(L_1) = \left\{ I \in \mathcal{MC}_2 : \forall p \in \widetilde{C}_I,\ V(p) \cap \left(\overline{V}(L_1) \setminus V(L_1) \right) \neq \emptyset \right\} \cup \left\{ I \in \mathcal{MC}_3 : \mathcal{F}_I \subseteq \overline{V}(L_1) \setminus V(L_1) \right\}$$

be the subset of nonrobust cover inequalities that will never be active for a path extended from label L_1. $\overline{V}(L_1) \setminus V(L_1)$ is the set of nodes that have not been visited in path $p(L_1)$ and cannot be visited in any extension of L_1. If this set intersects with all the paths defining a nonrobust p-cover inequality in $\mathcal{MC}_2$ or includes the set of nodes $\mathcal{F}_I$ in the case of a nonrobust q-cover inequality in $\mathcal{MC}_3$, then any extension of L_1 will never contribute to the inequality. Considering Example 2, the nonrobust p-cover inequality defined by the p-cover $C = \{3,4\}$ will never be active in a path that is extended from label L_2.

Furthermore, let

$$\varphi(L_2) = \{ I \in \mathcal{MC}_2 \cup \mathcal{MC}_3 : d(L_2) \geq \tau_I \} \cup \left\{ I \in \mathcal{MC}_2 : \exists p \in \widetilde{C}_I, V(p) \subseteq V(L_2) \right\} \cup \{ I \in \mathcal{MC}_3 : \exists (f_i, q_i) \in \mathcal{F}_I \times \mathcal{Q}_I, p(L_2) \in \Omega^s(f_i, q_i) \}$$

be the subset of nonrobust cover inequalities in $\mathcal{MC}_2 \cup \mathcal{MC}_3$ for which we know for sure that label L_2 will be extended into a path that will contribute to one of its nonrobust cover inequalities. If we now define $\theta(L_1, L_2) = \phi(V(L_1) \setminus V(L_2)) \setminus (\chi(L_1) \cup \varphi(L_2))$, we get the improved dominance criterion.

Proposition 6. *Label L_2 is dominated by label L_1 if*

1. $v(L_1) = v(L_2)$.
2. $c(L_1) - \sum_{I \in \theta(L_1, L_2)} v_I \leq c(L_2)$.
3. $t(L_1) \leq t(L_2)$.
4. $d(L_1) \leq d(L_2)$.
5. $\overline{V}(L_1) \subseteq \overline{V}(L_2)$.

To summarize, we have the choice between three possible dominance tests: the dominance of Proposition 2 but using the set $V(L)$ instead of $\overline{V}(L)$, the

dominance of Proposition 5, and the dominance of Proposition 6. We have tested all three possibilities; the best results were generated by using the dominance of Proposition 6. Therefore, we choose the dominance of Proposition 6 for our numerical experiments.

7.2. *ng*-Path Pricing

It is well known from the VRPTW that the ESPPRC can be very time consuming to solve, and for hard instances, a large fraction of the time spent by a BCP algorithm may be spent on solving the pricing subproblem. Consequently, several papers have considered relaxations of the ESPPRC. The most successful relaxation currently is the *ng*-path relaxation proposed in Baldacci et al. (2011b). In the *ng*-path relaxation, one defines a neighborhood N_i for each customer i; the neighborhood N_i controls which customers to remember as the path traverses node i. Earlier visits to nodes in N_i will be remembered, whereas visits to nodes from $V_c \setminus N_i$ are forgotten. If a visit to a node j is forgotten, it implies that the path may return to node j and thereby create a cycle. By selecting N_i wisely, one can make paths involving cycles rather unlikely.

It is easy to adapt the label-setting algorithm described in Section 7.1 to solve the *ng*-path problem. An *ng* label L contains the attributes $v(L), c(L), d(L)$, and $t(L)$ that are defined in the same way as for the ESPPRC. Furthermore, L contains a set $V^{ng}(L)$ that contains the set of nodes visited by the partial path $p(L)$ that are remembered at node $v(L)$. When extending a label L to a node i, one updates $V^{ng}(\cdot)$ as follows:

$$V^{ng}(L^{\text{new}}) = (V^{ng}(L) \cap N_i) \cup \{i\}.$$

If nonrobust cuts are not used, the following proposition defines a domination criterion for *ng*-paths.

Proposition 7. *Label L_2 is dominated by label L_1 if*

1. $v(L_1) = v(L_2)$.
2. $c(L_1) \leq c(L_2)$.
3. $t(L_1) \leq t(L_2)$.
4. $d(L_1) \leq d(L_2)$.
5. $V^{ng}(L_1) \subseteq V^{ng}(L_2)$.

Handling the k-, p-, and q-cover inequalities is similar to the ESPPRC case. Because k-cover inequalities are straightforward, we focus on the p- and q-cover inequalities. Given a set of p- and q-cover inequalities $\mathcal{MC}_2 \cup \mathcal{MC}_3$, we define $\theta^{ng}(L_1, L_2)$ to be the set of inequalities from $\mathcal{MC}_2 \cup \mathcal{MC}_3$ such that for each inequality $I \in \theta^{ng}(L_1, L_2)$ there exists an extension L of L_2 such that $p(L_2 \oplus L)$ does not contribute to the left-hand side of I, whereas $p(L_1 \oplus L)$ does. It may be difficult to define $\theta^{ng}(L_1, L_2)$ exactly, but a function $\bar{\theta}^{ng}(L_1, L_2)$ that for any pair of labels L_1, L_2 creates a superset of $\theta^{ng}(L_1, L_2)$ works as well; that is, $\bar{\theta}^{ng}$ should satisfy $\theta^{ng}(L_1, L_2) \subseteq \bar{\theta}^{ng}(L_1, L_2)$ for all L_1, L_2. With this we can define a domination criterion that is valid when nonrobust cuts are included.

Proposition 8. *Label L_2 is dominated by label L_1 if*

1. $v(L_1) = v(L_2)$.
2. $c(L_1) - \sum_{I \in \bar{\theta}^{ng}(L_1, L_2)} v_I \leq c(L_2)$.
3. $t(L_1) \leq t(L_2)$.
4. $d(L_1) \leq d(L_2)$.
5. $V^{ng}(L_1) \subseteq V^{ng}(L_2)$.

To compute $\bar{\theta}^{ng}(L_1, L_2)$, we extend the set of label attributes with a set $V(L)$ that contains all nodes visited by path $p(L)$, and we define two functions processPCover and processQCover in Algorithms 1 and 2. processPCover goes through the active p-cover cuts and returns the ones that belong to $\bar{\theta}^{ng}(L_1, L_2)$. In Algorithm 1, the symbol $\mathcal{P}(C_I)$ denotes the paths that occur on the left-hand side of p-cover inequality I, and $\tilde{C}_I$ is the set of trimmed paths that define inequality I (see definition of $\mathcal{P}(C)$ and $\tilde{C}$ in Section 5.3.3).

The check in line 4 of Algorithm 1 ensures that if $p(L_2)$ and all extensions $p(L_2)$ belong to $\mathcal{P}(C_I)$, then I will not be added to $\bar{\theta}^{ng}$. The check in line 6 of Algorithm 1 ensures that I only is added to $\bar{\theta}^{ng}$ if there is a chance that an extension of $p(L_1)$ will contribute to I, whereas the same extension of $p(L_2)$ potentially is not contributing. The same inequality may be added several times in line 7 of Algorithm 1, but set operations ensure that it only appears once in $\bar{\theta}^{ng}$.

Algorithm 1 (Find p-Cover Inequalities That Should Be Added to $\bar{\theta}^{ng}(L_1, L_2)$)

```
1: function processPCover(MC_2, L_1, L_2)
2:   θ̄^ng = ∅
3:   for (I ∈ MC_2) do
4:     if p(L_2) ∉ P(C_I), then
5:       for p ∈ C̃_I do
6:         if (V(L_1) \ V(L_2)) ∩ v(p) ≠ ∅), then
7:           θ̄^ng = θ̄^ng ∪ {I}
8:         end if
9:       end for
10:    end if
11:  end for
12: return θ̄^ng
13: end function
```

processQCover is similar to processPCover but handles q-cover inequalities. In the pseudocode, $\mathcal{F}_I$ and $\mathcal{Q}_I$ denote the set of nodes and demands that define inequality I, whereas f_i^I and q_i^I denote the ith element in $\mathcal{F}_I$ and $\mathcal{Q}_I$, respectively.

The checks in lines 4 and 6 of Algorithm 1 ensure that inequalities that any path extended from $p(L_2)$ will contribute to are not returned. The check in line 10 of Algorithm 1 ensures that I only is added to $\bar{\theta}^{ng}$ if there is a chance that an extension of $p(L_1)$ will

contribute to I, whereas the same extension of $p(L_2)$ potentially is not contributing.

With these functions, we can define

$$\bar{\theta}^{ng}(L_1, L_2) = \text{PROCESSPCOVER}(\mathcal{MC}_2, L_1, L_2) \cup \text{PROCESSQCOVER}(\mathcal{MC}_3, L_1, L_2).$$

The algorithm for solving the ng-path pricing problem is embedded in a bidirectional label-setting algorithm (see Righini and Salani 2006) to speed up computations.

Algorithm 2 (Find q-Cover Inequalities That Should Be Added to $\bar{\theta}^{ng}(L_1, L_2)$)

function PROCESSQCOVER($\mathcal{MC}_3, L_1, L_2$)
  $\bar{\theta}^{ng} = \emptyset$
  for $I \in \mathcal{MC}_3$ **do**
    if $d(L_2) < \max\{q : q \in \mathcal{Q}_I\}$, **then**
      for $i = 1$ to $|\mathcal{F}_I|$ **do**
        if $((f_i^I \in V(L_2)) \wedge (d(L_2) > q_i^I))$, **then**
          skip to next $I \in \mathcal{MC}_3$
        end if
      end for
      if $(V(L_1) \setminus V(L_2)) \cap \mathcal{F}_I) \neq \emptyset$, **then**
        $\bar{\theta}^{ng} = \bar{\theta}^{ng} \cup \{I\}$
      end if
    end if
  end for
return $\bar{\theta}^{ng}$
end function

8. Branching

The branch-and-bound tree is explored using a best-bound strategy. First, the algorithm branches on the arc variables x_{ij}^s. It looks for pairs $(i, j), i, j \in V_c$ and shifts $s \in S$ such that $x_{ij}^{s*} + x_{ji}^{s*}$ is close to 0.5 (x^* is the current fractional solution expressed in the arc variables) and imposes two branches $x_{ij}^s + x_{ji}^s \leq 0$ and $x_{ij}^s + x_{ji}^s \geq 1$. If $x_{ij}^{s*} + x_{ji}^{s*}$ is an integer for all pairs $(i, j), i, j \in V_c$ and shifts $s \in S$, then the algorithm looks for an arc $(i, j) \in A$ and a shift $s \in S$ for which x_{ij}^{s*} is fractional and branches on that instead. Strong branching is used; that is, the impact of branching on several candidates is investigated every time a branching decision has to be made. For each branch candidate, we estimate the lower bound in the two child nodes by solving the associated LP relaxation using a quick pricing heuristic. The branch that maximizes the primal bound in the weakest of the two child nodes is chosen. We consider 30 branch candidates in the first 20 nodes of the branch-and-bound tree and 20 candidates in the rest.

9. Computational Results

The BCP algorithm is implemented in C++ on an Intel Xeon central processing unit (CPU), 2.67 GHz. For all experiments, we use a time limit of 1 hour. The LP solver CLP from the open-source framework COIN (COIN CLP 2011) is used to solve the LP relaxation of the master problem. For our numerical study, we use the well-known Solomon data sets (Solomon 1987). Instances with 25, 50, and 100 customers are tested. For each instance size, six categories of instances are tested, $R1, R2, C1, C2, RC1$, and $RC2$, according to the geographic distribution and the tightness of time windows. The geographic distribution of the customers is randomly generated for sets $R1$ and $R2$, are clustered for sets $C1$ and $C2$, and are a mix of random and clustered for sets $RC1$ and $RC2$. The time windows are tight in sets $R1, C1$, and $RC1$ and wide in sets $R2, C2$, and $RC2$, which allow more customers per route. In this paper, the Solomon instances are represented using the notation DTm. D is the geographic distribution of the customers, which can be R, C, or RC. T is the instance type, which can be either 1 (instances with tight time windows) or 2 (instances with wide time windows), and m denotes the number of instance. For all instances, we consider three shifts with equal loading capacity, which is calculated as

$$\rho \frac{\sum_{i \in V_c} d_i}{3},$$

where $\rho \in \{1.05, 1.2, 1.5\}$. Furthermore, the depot's operating period is divided into three equally long periods with length $\frac{b_{n+1}}{3}$ such that each period is assigned to a different shift. We note that our instances do not cover situations in which shifts have different operating period lengths and loading capacities. The developed algorithm can, however, deal with these types of instances. Furthermore, the real-life application that inspired us to research this problem has three shifts with equal operating period lengths.

We organize the computational experiments in two phases. First, we experiment with different route relaxations in the pricing problem to determine the route relaxation that is most efficient for our problem. Second, and once the route relaxation to be used in the pricing problem is decided on, we run all the instances using six different algorithm settings (see Table 4). Algorithm $\mathcal{A}_1$ is the basic algorithm in which we do not include any of the valid inequalities. Algorithm $\mathcal{A}_2$ implements (lifted) robust cover inequalities ($\mathcal{CC}$) but none of the nonrobust cover inequalities. Algorithm $\mathcal{A}_3$ implements, in addition to $\mathcal{CC}$, classic VRP valid inequalities, that is, capacity and two-path inequalities ($\mathcal{X}$). Algorithm $\mathcal{A}_4$ implements, in addition to $\mathcal{CC}$, nonrobust k-cover inequalities ($\mathcal{MC}_1$). Furthermore, algorithm $\mathcal{A}_5$ supports nonrobust p-cover inequalities ($\mathcal{MC}_2$), and algorithm $\mathcal{A}_6$ supports nonrobust q-cover inequalities ($\mathcal{MC}_3$). We also tested a configuration that implements all cover inequalities, but no clear performance improvements were observed, and therefore, the results are not reported. At any point during the search, we limit the number of

Table 4. Algorithm Overview

	$\mathscr{CC}$	$\mathscr{R}$	$\mathscr{MC}_1$	$\mathscr{MC}_2$	$\mathscr{MC}_3$
$\mathscr{A}_1$					
$\mathscr{A}_2$	✓				
$\mathscr{A}_3$	✓	✓			
$\mathscr{A}_4$	✓	✓	✓		
$\mathscr{A}_5$	✓	✓	✓	✓	
$\mathscr{A}_6$	✓	✓	✓		✓

active nonrobust cover inequalities in the master problem to 50. The algorithm has a buffer that is used to track the number of inequalities already added at any point during the search. The buffer's capacity is set to 50. We note that during execution of the algorithm, inequalities that are deemed to be inefficient are deleted, implying that new inequalities can be added instead as some capacity is freed in the buffer. The deleted nonrobust cuts are lost and are regenerated if it turns out to be violated again. The nonrobust cover inequalities are global inequalities, meaning that they are valid for all problems solved in each node of the branch-and-bound tree.

9.1. Comparison of the Different Pricing Problems

In the first phase of the computational experiments, we run different versions of the algorithm using different route relaxations in the pricing problem. We experiment with four different pricing algorithms: a pricing problem with elementary paths (denoted Elem) and a pricing problem with *ng*-paths. For the pricing problem with *ng*-paths, we experiment with three different *ng*-neighborhood sizes (i.e., 5, 10, and 15). We denote the corresponding pricing problems NG-5, NG-10, and NG-15, respectively. We test the different route relaxations on instances with 25 and 100 customers and $\rho = 1.2$. Table 5 reports results for each algorithm and for each pricing problem configuration. We report the average running time ("$\bar{t}$[s]") over instances that are solved by all algorithm configurations and the number of instances solved to optimality ("#Opt") by each algorithm configuration. From the highlighted results (in bold), NG-10 and NG-15 seem to provide the best performances. By further looking at the detailed results, it turns out that NG-10 is able to find a lower bound more often than NG-15. Therefore, we decide to choose NG-10 as our pricing problem for the remaining computational experiments.

9.2. Impact of the Nonrobust *q*-Cover Inequalities Separation Procedure

In the remaining computational experiments, all the algorithms implement NG-10 as the pricing problem. In Table 6, we compare the performance of algorithm $\mathscr{A}_6$ when the exact separation procedure is activated (i.e., solving the MIP (35)–(40)) after the heuristic procedure fails finding any violated nonrobust *q*-cover inequalities against the case in which only the heuristic separation procedure is used. We use CPLEX to solve the nonrobust *q*-cover inequalities separation problem (35)–(40). The first column ("Inst.") indicates the name of the instance. The columns denoted as "t[s]" show the time (in seconds) spent to solve an instance. The columns denoted as "LBr" show the lower bounds in the root node. In the columns labeled "Tree," we report the size of the branching trees. The columns denoted as "LB_{gap}" and "t_{gap}" show, respectively, by how much the lower bound worsens and how much CPU time is gained by applying only the heuristic separation. We report results for the instance with 25 customers and $\rho = 1.05$, for which both implementations could find a lower bound and observe that although, in many cases, the lower bound in the root node of the branch-and-bound tree improves slightly (0.05% on average) and the size of the branch-and-bound tree decreases when the exact procedure is activated, the runtime increases (by 7.1% on average). Therefore, in all tables that follow, we only report results for an $\mathscr{A}_6$ algorithm that supports only the heuristic separation procedure.

Table 5. Impact of Route Relaxation ($\rho = 1.2$)

$\lvert V_c\rvert$		25		100	
Algorithm	Relaxation	$\bar{t}$[s]	#Opt	$\bar{t}$[s]	#Opt
$\mathscr{A}_1$	Elem	308	47	1,247	8
	NG-15	342	**49**	**1,012**	**9**
	NG-10	**261**	48	1,203	**9**
	NG-5	313	**49**	1,017	**9**
$\mathscr{A}_2$	Elem	293	47	1,061	9
	NG-15	343	**49**	**928**	**10**
	NG-10	**284**	**49**	1,000	**10**
	NG-5	320	**49**	982	9
$\mathscr{A}_3$	Elem	281	47	1,171	9
	NG-15	339	49	**928**	**11**
	NG-10	354	**50**	962	**11**
	NG-5	**313**	49	1,093	**11**
$\mathscr{A}_4$	Elem	271	49	1,300	10
	NG-15	315	**51**	**914**	**11**
	NG-10	**264**	**51**	989	**11**
	NG-5	278	**51**	955	10
$\mathscr{A}_5$	Elem	213	49	1,325	11
	NG-15	**208**	50	1,075	**12**
	NG-10	**208**	**51**	**950**	11
	NG-5	216	**51**	998	10
$\mathscr{A}_6$	Elem	190	49	1,275	**12**
	NG-15	219	**51**	**851**	**12**
	NG-10	**173**	**51**	1,017	**12**
	NG-5	178	**51**	895	**12**

Table 6. Exact vs. Heuristic Separation in $\mathcal{A}_6$

	Heuristic separation			Exact separation				
Inst.	t[s]	LBr	Tree	t[s]	LBr	Tree	t_{gap}(%)	LB_{gap}(%)
R101	16	697.8	214	21	698.9	152	22.0	–0.15
R102	18	573.8	146	24	573.8	138	25.1	0.00
R103	105	485.6	592	119	485.6	476	11.4	0.00
R104	287	467.7	966	265	468.9	770	–8.1	–0.26
R105	18	582.4	148	37	584.9	168	50.6	-0.43
R106	21	477.4	92	35	477.4	108	38.8	0.00
R107	21	442.3	46	31	442.6	54	33.2	–0.08
R108	78	421.1	130	71	421.1	102	–10.3	0.00
R109	94	470.8	474	118	471.2	480	20.9	–0.10
R110	121	446.3	332	134	446.3	282	10.2	0.00
R111	48	455.7	134	63	455.7	132	24.3	0.00
R112	230	404.1	378	250	404.1	356	8.0	0.00
C101	41	263.9	158	46	263.9	134	11.7	–0.01
C102	42	246.0	68	48	246.0	66	12.8	0.00
C103	120	236.9	90	109	236.9	82	–9.6	0.00
C104	970	231.2	582	1,110	231.2	640	12.6	0.00
C105	16	246.6	46	18	246.6	42	13.0	0.00
C106	50	263.9	180	67	263.9	182	25.5	0.00
C107	23	244.9	60	44	244.9	88	46.4	0.00
C108	154	243.9	214	105	244.9	164	–46.6	–0.39
C109	529	235.7	440	373	235.7	352	–41.8	0.00
RC101	—	546.2	14,444	—	546.2	18,568	—	0.00
RC102	631	406.0	1,156	747	406.0	1,450	15.5	0.00
RC103	565	384.1	934	537	384.1	870	–5.1	0.00
RC104	—	346.6	1,810	—	346.6	2,298	—	0.00
RC105	383	501.0	1,884	528	501.0	1,934	27.4	0.00
RC106	77	451.2	148	88	451.2	148	12.5	0.00
RC107	2,534	407.6	2,444	2,251	409.1	2,460	–12.6	–0.36
RC108	—	347.5	1,876	—	347.5	2,140	—	0.00
R201	15	490.6	54	22	490.6	60	30.5	0.00
R202	28	422.1	48	37	425.4	48	24.2	–0.77
R203	13	399.0	6	15	399.0	6	9.3	0.00
R204	30	369.6	8	32	369.6	8	6.0	0.00
R205	100	404.9	146	109	404.9	154	8.1	0.00
R206	22	380.2	20	24	380.2	20	6.5	0.00
R207	101	368.9	46	105	368.9	46	3.6	0.00
R208	150	361.2	60	153	361.2	60	2.2	0.00
R209	5	381.1	2	5	381.1	2	5.8	0.00
R210	51	414.1	22	57	414.1	24	10.3	0.00
R211	4	364.9	0	4	364.9	0	0.5	0.00
C201	16	287.2	44	18	287.2	44	11.0	0.00
C202	36	280.2	30	39	280.2	30	5.8	0.00
C203	159	277.4	38	169	277.4	38	6.1	0.00
C204	570	274.1	90	562	274.1	90	–1.4	0.00
C205	27	286.4	44	28	286.4	44	6.3	0.00
C206	37	283.9	44	40	283.9	44	6.2	0.00
C207	115	282.7	16	119	282.7	16	3.1	0.00
C208	60	283.0	30	64	283.0	30	6.0	0.00
RC201	1,460	459.3	1,820	672	459.3	994	–117.4	0.00
RC202	822	347.4	230	1,032	347.4	236	20.3	0.00
RC203	—	340.5	346	—	340.5	332	—	0.00
RC204	—	309.2	134	—	309.2	134	—	0.00
RC205	270	347.7	132	293	347.7	150	7.8	0.00
RC206	473	346.7	290	471	346.7	284	–0.5	0.00
RC207	—	313.4	540	—	313.4	556	—	0.00
RC208	—	304.5	72	—	304.5	74	—	0.00
Average							7.1	–0.05

Table 7. Impact of Shift Loading Capacity

ρ	1.05			1.2			1.5		
$\lvert V_c\rvert$	25	50	100	25	50	100	25	50	100
#Opt	50	32	15	52	33	12	50	25	19
$\bar{t}$[s]	42.3	9,518.6	13,640.6	3,745.6	8,946.3	12,922.2	3,531.6	5,768.1	12,416.7
$\overline{\text{Cost}}$	50.8	183.8	225.6	38.5	149.7	130.2	29.8	79.0	124.5

9.3. General Findings

As expected, adding shift loading capacities to the vehicle routing problem with time windows adds to its complexity. However, it is remarkable how complicated the resulting problem (i.e., the VRPTWS) becomes. This complexity is reflected by the solution running times and the large size of the branching trees, especially when shift loading capacities are binding (e.g., instances RC101, RC104, and RC108 for $\rho = 1.05$ and 25 customers; see the online appendix). Furthermore, the shift loading capacities have a significant impact on the costs. Table 7 shows the impact, for different instance sizes, of shift loading capacity on solution complexity and cost. The row "#Opt" shows the number of instances that could be solved to proven optimality by at least one of the algorithms. The rows "$\bar{t}$[s]" and "$\overline{\text{Cost}}$" show, for each instance size, the average running time and average solution value over all instances solved to proven optimality by at least one algorithm, respectively. Decreasing ρ clearly results, on average, in an increase in cost. In fact, tightening shift loading capacity enforces planning customers on later shifts. Ensuring route feasibility with regard to time windows comes at the expense of traveling cost. Furthermore, tightening shift loading capacity results in more difficult instances because running times increase on average. Table 8 reports details for the number of instances solved to proven optimality by each of the algorithms. Clearly, in most cases, algorithms implementing nonrobust cover inequalities perform better. However, in some cases, there are a few instances that can be solved by algorithms not supporting nonrobust cover inequalities but not by algorithms supporting nonrobust cover inequalities (e.g., instances RC203 for $\rho = 1.05$ and 25 customers and C103 and C106 for $\rho = 1.05$ and 100 customers; see the online appendix).

Table 8. Number of Instances Solved to Optimality by Each Algorithm

$\lvert V_c\rvert$	25			50			100		
ρ	1.05	1.2	1.5	1.05	1.2	1.5	1.05	1.2	1.5
$\mathcal{A}_1$	46	48	46	18	21	24	9	9	13
$\mathcal{A}_2$	46	49	46	18	22	24	10	10	14
$\mathcal{A}_3$	46	50	46	19	20	23	14	11	18
$\mathcal{A}_4$	47	51	47	19	25	22	11	11	18
$\mathcal{A}_5$	48	51	48	24	29	24	12	11	19
$\mathcal{A}_6$	49	51	50	30	32	25	12	12	19

Table 9. Aggregate Comparison Between Pricing Algorithms with Different Cuts

Algorithm	$\overline{\text{LBr}}$	$\overline{\text{LBb}}$	$\bar{t}$[s]	$\overline{\text{Tree}}$
$\mathcal{A}_1$	652.8	667.5	406.6	451.6
$\mathcal{A}_2$	652.8	667.5	384.6	374.4
$\mathcal{A}_3$	656.4	668.3	364.6	349.1
$\mathcal{A}_4$	657.4	669.2	350.7	297.0
$\mathcal{A}_5$	657.5	669.5	302.4	185.4
$\mathcal{A}_6$	658.3	670.2	272.1	151.1

9.4. Impact of the Valid Inequalities

Table 9 provides a comparison of all the implemented algorithms. The columns "$\overline{\text{LBr}}$" and "$\overline{\text{LBb}}$" indicate, respectively, the average of the root lower bound and the average of the best lower bound of the instances for which all algorithms are able to produce a lower bound. Moreover, the average computation time (in seconds) and the average number of nodes in the branch-and-bound trees over all the instances solved to optimality by all algorithms are reported in the columns "$\bar{t}$[s]" and "$\overline{\text{Tree}}$," respectively. Again, algorithms $\mathcal{A}_5$ and $\mathcal{A}_6$ outperform the other algorithms. We note that algorithm $\mathcal{A}_6$ outperforms $\mathcal{A}_5$. Table 10 shows that the root node of only very few instances with 100 customers could not be solved by

Table 10 Number of Instances with non solved root node

		R instances						C instances						RC instances					
$\lvert V\text{c}\rvert$	ρ	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$	$\mathcal{A}_5$	$\mathcal{A}_6$	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$	$\mathcal{A}_5$	$\mathcal{A}_6$	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$	$\mathcal{A}_5$	$\mathcal{A}_5$
100	1.2						1	1	1	1	1	1	1	1	1	1	1	1	1
	1.5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

the algorithms. Detailed results are reported in the online appendix.

10. Conclusions

In real life, loading vehicles is constrained by the shift loading capacities at the warehouses. In this paper, we explicitly consider shift loading capacity, which, in our context, leads to the vehicle routing problem with time windows and shifts. Limited shift loading capacities are modeled by knapsack inequalities, where the knapsacks are the shifts and the items to pack are either the customers or the paths. Inspired by valid inequalities for the knapsack problem, we developed tailored robust and new nonrobust cover inequalities. Nonrobust cover inequalities defined on the master variables are clearly stronger but significantly complicate the pricing subproblem, which is consistent with previous research on VRP algorithms that include nonrobust inequalities. They may be applied to other problems and benefit from ideas introduced in previous research, such as the introduction of a weakened version of the inequalities based on the limited-memory techniques (e.g., Pecin et al. 2017a, b) proven useful for other families of nonrobust inequalities such as the subset–row inequalities. However, further research and implementations are needed to investigate the value of such techniques for the newly introduced nonrobust cover inequalities and to show their value when applied in another context than the VRPTWS. We succeed in handling the nonrobust inequalities in an efficient way and show their value in extended computational experiments. The algorithm can handle some instances with up to 100 customers and three shifts.

References

Balas E (1975) Facets of the knapsack polytope. *Math. Programming* 8(1):146–164.

Baldacci R, Mingozzi A (2009) A unified exact method for solving different classes of vehicle routing problems. *Math. Programming* 120(2):347–380.

Baldacci R, Mingozzi A, Calvo RW (2011a) An exact method for the capacitated location-routing problem. *Oper. Res.* 59(5):1284–1296.

Baldacci R, Mingozzi A, Roberti R (2011b) New route relaxation and pricing strategies for the vehicle routing problem. *Oper. Res.* 59(5):1269–1283.

Baldacci R, Mingozzi A, Roberti R (2012) Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *Eur. J. Oper. Res.* 218(1):1–6.

Baldacci R, Mingozzi A, Roberti R, Calvo RW (2013) An exact algorithm for the two-echelon capacitated vehicle routing problem. *Oper. Res.* 61(2):298–314.

Bettinelli A, Ceseli A, Righini G (2011) A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows. *Transportation Res. Part C* 19(5): 723–740.

Bräysy O, Gendreau M (2005a) Vehicle routing problem with time windows, part I: Route construction and local search algorithms. *Transportation Sci.* 39(1):104–118.

Bräysy O, Gendreau M (2005b) Vehicle routing problem with time windows, part II: Metaheuristics. *Transportation Sci.* 39(1):119–139.

Chabrier A (2006) Vehicle routing problem with elementary shortest path based column generation. *Comput. Oper. Res.* 33(10):2972–2990.

Chao I, Golden BL, Wasil E (1995) An improved heuristic for the period vehicle routing problem. *Networks* 26(1):25–44.

COIN CLP (2011) COIN-OR linear programming solver. Accessed January 1, 2014, https://projects.coin-or.org/Clp.

Contardo C, Martinelli R (2014) A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optim.* 12(2014) 129–146.

Cook W, Rich JL (1999) A parallel cutting plane algorithm for the vehicle routing problem with time windows. Technical Report TR99-04, Rice University, Houston.

Cordeau JF, Gendreau M, Laporte G (1997) A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks* 30(2):105–119.

Cordeau JF, Maischberger M (2012) A parallel iterated tabu search heuristic for vehicle routing problems. *Comput. Oper. Res.* 39(9): 2033–2050.

Desaulniers G, Desrosiers J, Solomon MM (2005) *Column Generation* (Springer, New York).

Desaulniers G, Lessard F, Hadjar A (2008) Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Sci.* 42(3): 387–404.

Desaulniers G, Madsen OBG, Ropke S (2014) The vehicle routing problem with time windows. Toth P, Vigo D, eds. *Vehicle Routing: Problems, Methods, and Applications*, 2nd ed. (SIAM, Philadelphia), 119–159.

Desrochers M, Desrosiers J, Solomon M (1992) A new optimization algorithm for the vehicle routing problem with time windows. *Oper. Res.* 40(2):342–354.

Feillet D, Dejax P, Gendreau M, Gueguen C (2004) An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks* 44(3):216–229.

Gendreau M, Tarantilis CD (2010) Solving large-scale vehicle routing problems with time windows: The state of the art. Technical Report 2010-04, CIRRELT, Montreal.

Gromicho J, van Hoorn JJ, Kok AL, Schutten JMJ (2012) Vehicle routing with restricted loading capacities. Beta working paper, Eindhoven University of Technology, Eindhoven, Netherlands.

Gu Z, Nemhauser GL, Savelbergh M WP (1998) Lifted cover inequalities for 0-1 integer programs: Computation. *INFORMS J. Comput.* 10(4):427–437.

Hemmelmayr VC, Doerner KF, Hartl RF (2009) A variable neighborhood search heuristic for periodic routing problems. *Eur. J. Oper. Res.* 195(3):791–802.

Ho W, Ho GTS, Ji P, Lau HCW (2008) A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Engrg Appl. Artificial Intelligence* 21(4):548–557.

Irnich S, Villeneuve D (2006) The shortest path problem with resource constraints and *k*-cycle elimination for $k \geq 3$. *INFORMS J. Comput.* 18(3):391–406.

Jepsen M, Petersen B, Spoorendonk S, Pisinger D (2008) Subset-row inequalities applied to the vehicle-routing problem with time windows. *Oper. Res.* 56(2):497–511.

Kallehauge B (2008) Formulations and exact algorithms for the vehicle routing problem with time windows. *Comput. Oper. Res.* 35(7):2307–2330.

Kaparis K, Letchford AN (2008) Local and global lifted cover inequalities for the 0-1 multidimensional knapsack problem. *Eur. J. Oper. Res.* 186(2008):91–103.

Kohl N, Desrosiers J, Madsen OBG, Solomon MM, Soumis F (1999) 2-path cuts for the vehicle routing problem with time windows. *Transportation Sci.* 33(1):101–116.

Laporte G (1992) The vehicle routing problem: An overview of exact and approximate algorithms. *Eur. J. Oper. Res.* 59(3):345–358.

Laporte G (2007) What you should know about the vehicle routing problem. *Naval Res. Logist.* 54(8):811–819.

Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. *Oper. Res.* 53(6):1007–1023.

Lysgaard J (2003) CVRPSEP: A package of separation routines for the capacitated vehicle routing problem. Working Paper 03–04, Aarhus School of Business, Aarhus, Denmark.

Lysgaard J, Letchford AN, Eglese RW (2004) A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Math. Programming* 100(2):423–445.

Mourgaya M, Vanderbeck F (2007) Column generation based heuristic for tactical planning in multi-period vehicle routing. *Eur. J. Oper. Res.* 183(3):1028–1041.

Muter I, Cordeau JF, Laporte G (2014) A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes a branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes. *Transportation Sci.* 48(3): 425–441.

Pecin D, Contardo C, Desaulniers G, Uchoa E (2017a) New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS J. Comput.* 29(3):489–502.

Pecin D, Pessoa A, Poggi M, Uchoa E (2017b) Improved branch-cut-and-price for capacitated vehicle routing. *Math. Programming Comput.* 9(1):61–100.

Pisinger D (1997) A minimal algorithm for the 0-1 knapsack problem. *Oper. Res.* 45(5):758–767.

Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. *Comput. Oper. Res.* 34(8):2403–2435.

Prodhon C, Prins C (2014) A survey of recent research on location-routing problems. *Eur. J. Oper. Res.* 238(1):1–17.

Ren Y, Dessouky M, Ordóñez F (2010) The multi-shift vehicle routing problem with overtime. *Comput. Oper. Res.* 37(11):1987–1998.

Righini G, Salani M (2006) Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optim.* 3(3):255–273.

Righini G, Salani M (2008) New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks* 51(3):155–170.

Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* 35(2): 254–265.

Toth P, Vigo D (2014) *Vehicle Routing: Problems, Methods, and Applications*, vol. 18 (SIAM, Philadelphia).

Vidal T, Crainic TG, Gendreau M, Lahrichi N, Rei W (2012) A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Oper. Res.* 60(3):611–624.

Wolsey LA (1975) Faces for linear inequalities in 0-1 variables. *Math. Programming* 8(1):165–178.

Wu TH, Low C, Bai J-W (2002) Heuristic solutions to multi-depot location-routing problems. *Comput. Oper. Res.* 29(10):1393–1415.